EP 23110 A

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 738 982 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
23.10.1996 Bulletin 1996/43

(51) Int Cl.6: **G06F 17/22**

(21) Application number: **96302507.7**

(22) Date of filing: **10.04.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **20.04.1995 US 425164**

(71) Applicant: **NCR INTERNATIONAL INC.**
Dayton, Ohio 45479 (US)

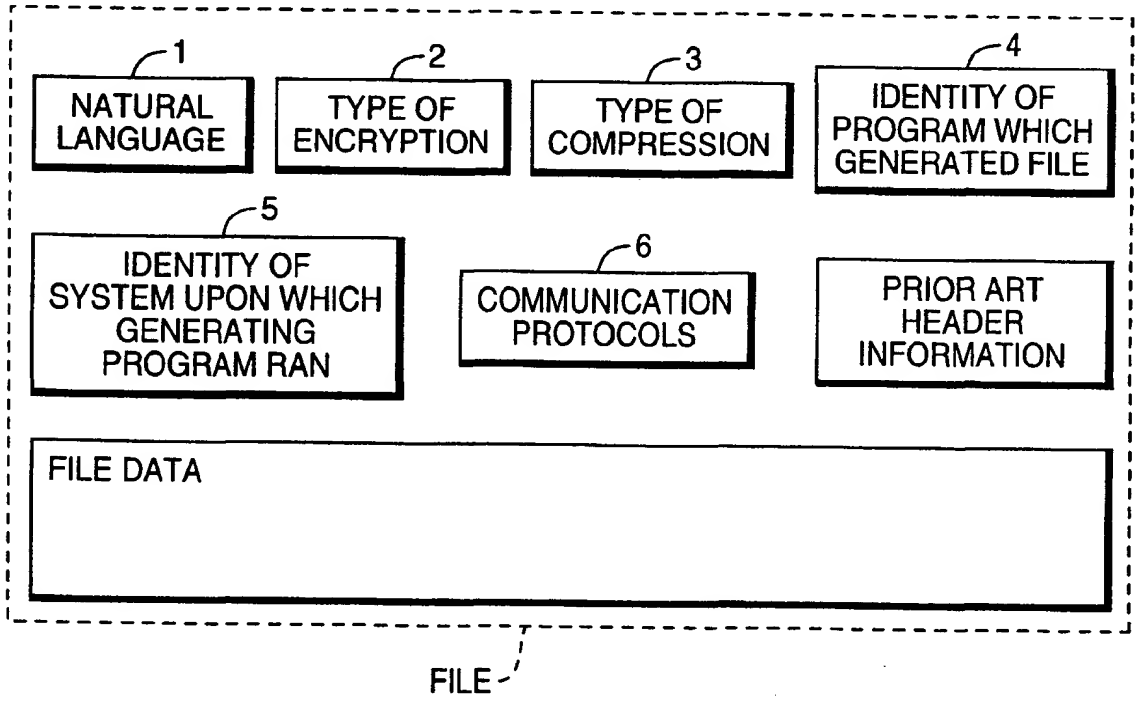
(72) Inventor: **Siefert, David M.**
Englewood, OH 45322 (US)

(74) Representative: **Robinson, Robert George**
International Intellectual Property Department,
NCR LIMITED,
915 High Road,
North Finchley
London N12 8QJ (GB)

(54) Method for generating a computer file

(57) The invention relates to means for generating a computer file comprising a data field and a header. The invention is characterized in that in the header is stored identification information other than the information required in the presentation of any data stored in the data field on a computer display screen.

FIG. 3



EP 0 738 982 A2

Printed by Jouve, 75001 PARIS (FR)

Description

The present invention relates to means for generating a computer file.

Computer software generates files, which generally take the form of digital representations of information. For example, when a user stores a document using word-processing software, a file is generated, which contains the document.

The software generally adds a "header" to the document itself. An example will illustrate.

Assume that a user generates the following document:

When buying coconuts, make sure that they are crack-free and have no mould on them. Shake them to make sure that they are heavy with water. Now hold coconut in one hand over a sink and hit it around the center with the claw end of a hammer.

When this document is stored as a binary file, each character (ie, the "W", "h", "e", "n", and so on) is assigned an eight-bit code. One type of assignment is shown in Figure 1, which illustrates a version of ASCII code. (As stated above, the file is generally stored in binary format. Hexadecimal, not binary, representation is used in Figure 1, for simplicity.)

When this file is stored in the normal manner by a word-processing program, a header is added, as shown in Figure 2. The particular header shown was added by the software package known as WordPerfect, available from WordPerfect Corporation, Orem, Utah. The header contains data such as the following:

Identification of the generating software ("WPC" in the header);
Identification of the printer selected by the user during generation of the document ("HP LaserJet Series II");
Identification of the font selected by the user ("Courier");
Identification of the pitch of the font ("10 pitch");

as well as other information. Therefore, from one perspective, a computer file contains two types of information: (1) data created by the user and (2) header information, created by software operated by the user.

An object of the invention is to provide a means for producing file having a structure which contains information which can be used by apparatus which processes the file, so as to increase the ease of using the file in different computer systems.

According to a first aspect of the present invention there is provided a means for generating a computer file, characterized in that said file comprises a data field and a header in which is stored identification information other than the information required in the presentation of any data stored in said data field on a computer display screen.

According to a second aspect of the present invention there is provided a method of processing a computer file, comprising the following steps:

a) maintaining, in association with the file, information indicative of two or more of the following:

i) natural language of the file;
ii) type of encryption of the file;
iii) type of compression of the file;
iv) identity of the program which generated the file;
v) identity of the operating system upon which the generating program ran; and
vi) communication protocol required by the file;

b) reading said information and calling respective programs which process the file in accordance with the information.

Embodiments of the present invention will now be described by way of example, with reference to the accompanying drawings, in which:-

Fig. 1 illustrates a hex dump of a text file, together with hex character equivalents;
Fig. 2 illustrates a hex dump of a word-processing file, together with hex character equivalents;
Fig. 3 illustrates one form of the invention;
Fig. 4 illustrates apparatus which can use the information shown in Fig. 3 for processing the FILE DATA;
Fig. 5 illustrates how certain information of Fig. 3 can be stored physically within a file;
Fig. 6 illustrates another form of the invention; and
Fig. 7 illustrates yet another form of the invention.

Figure 3 shows one form of a computer file produced by a means in accordance with the invention. Six blocks, labeled 1 through 6, have been added to a normal, prior-art, file which includes two components: (1) PRIOR ART HEADER INFORMATION and (2) FILE DATA (also called user data).

The PRIOR ART HEADER INFORMATION corresponds, in principle, to the header shown in Figure 2, and is generated by the software used to create the file. The user of the software, in general, is not involved with, and is not interested in, this HEADER.

The FILE DATA contains the information generated by the user, such as the phrase "When buying coconuts . . ." indicated in Figure 2. The FILE DATA is the information generated by the user.

For example, in the case of a word-processing program, the FILE DATA corresponds to the document generated by the user. As another example, in the case of a drafting program, the FILE DATA corresponds to the drawing generated by the user.

(In practice, a "system header" is also created, but is not shown. The system header is generated by, and

used by, the operating system, which is the physical agent which actually writes the bits, which make up the file, onto disc.)

Six Types of Added Information

The six added blocks of Figure 3 contain the following information. Block 1 contains information as to the natural language, if any, of the FILE DATA. Natural language refers to a human language, such as English or French, rather than another type of language, such as FORTRAN.

Block 2 contains information about the type of encryption, if any, which the FILE DATA has experienced, or should experience.

Block 3 contains information about the type of compression, if any, which the FILE data has experienced, or should experience.

Block 4 contains information about the identity of the program which generated the FILE DATA, such as the name of the particular program generating the file, and the release number, or version number, of the program. (Programs continually undergo revision, and the revisions themselves are continually revised.)

Block 5 contains information about the system upon which the generating program, identified in block 4, ran. Examples of different systems are Macintosh, IBM PC, Sun Workstation, Unix, etc. This information is relevant to the system header, discussed below. Block 5 may also contain information about the operating system upon which the generating program ran, such as DOS version 6.0.

Block 6 contains information needed by communication protocols, which may be used subsequently to transmit the file electronically from one location to another. As an example, some communications protocols transmit only seven-bit information. With such protocols, eight-bit data, such as compiled computer programs and some type of bit-mapped data, cannot be transmitted. Block 6 in Figure 3 would indicate the type of data (eight-bit or seven-bit) contained in the file, and thus indicate which protocols are needed.

Some examples of protocols are the commercial standards ISDN, TCP/IP, and ATM.

How Six Types are Used

The information contained within the six blocks is used by the systems shown in Figure 4. System 10 reads block 1, and calls an appropriate language translation program. For example, if the language indicated in block 1 is German, then system 10 knows that one of the translation programs in group 12 will be used. The translation will be from German, to another language. If the language is English, then system 10 knows that one of the translation programs in group 14 will be used. (The particular program in each group which will be selected will depend upon the language into which the file is to be translated, and is not a concern of the present invention.)

System 20 reads the information of block 2, and calls an appropriate encryption, or de-encryption, program.

System 30 reads the information of block 3, and calls an appropriate compression, or de-compression, program.

System 40 reads the information of block 4, and calls an appropriate table which contains relevant characteristics of the generating program. For example, if the generating program is WordPerfect, system 40 calls a table containing information needed to translate, or use, the header information shown in Figure 2.

System 50 reads the information of block 5, and calls an appropriate table which contains relevant characteristics of the system upon which the generating program ran. This table contains, among other things, information about the system header of the file.

System 60 reads the information of block 6, and responds appropriately.

How to Store the Six Types of Data

How the data of the six blocks is associated with the prior-art file (ie, the FILE DATA and PRIOR ART HEADER INFORMATION of Figure 3) will now be discussed.

First Approach

The six blocks can be added to a file as shown in Figure 5, which shows six added fields, labeled ADDED INFORMATION. Each horizontal line in the fields indicates one character. Each field may be identified by a code, such as NL for Natural Language, EN for Encryption, and so on.

However, it is possible that adding information to the header of a file in this manner will disrupt operation of the generating program. For example, a word-processing program may expect a specific number, and type, of fields within the HEADER. If the program sees added fields, it may not recognize the file, and issue an error warning.

Consequently, after the six fields have been added, it is possible that programs intended to retrieve and process the FILE DATA must be specifically re-designed to negotiate the six added fields. Designing programs to accomplish this is within normal programmer's skill, but it is possible that many existing programs are not so designed. It is also possible that software developers may be resistant to undertake such re-design.

Second Approach

To avoid this problem, the header of the generating program can be modified to contain an empty region into which a user can insert data, and which the generating program does not use. For example, as shown in Figure

6, a file contains a SYSTEM HEADER and a header generated by the generating program, labeled APPLICATION PROGRAM HEADER, as usual. In addition, contained within the APPLICATION PROGRAM HEADER are contained two delimiters, such as AAAA and ZZZZ, as indicated. The user is allowed to insert data between these delimiters. In Figure 6, the six blocks of Figure 3 have been inserted.

Under this approach, the application program is designed to ignore information contained between these delimiters. How to design the programs to accomplish this is known in the art, once the idea is suggested.

For the invention to use such a file structure, the invention reads the file, in a manner known in the art, and looks for the proper delimiter, which is AAAA in this case. The invention then looks for the NATURAL LANGUAGE block, as by searching for the code NL (see Figure 5). When this code is found, the invention then reads a next predetermined number of subsequent characters to ascertain the natural language.

Next, the invention looks for the encryption code, EN, in a similar manner, and proceeds until the delimiter ZZZZ is found.

Thus, under this approach, a space is reserved within a header of a file (or elsewhere) into which a fourth party can insert information. The fourth party is, in this case, software associated with the invention.

The party is "fourth" because three other "parties" are involved in inserting data into a file: The application program is considered one party, because it generates the APPLICATION PROGRAM HEADER shown in Figure 6. The user is considered a second party, because the user generates the FILE DATA. The operating system is considered a third party, because it inserts a system header (not shown herein).

Third Approach

It may not be feasible to expect manufacturers of application programs to allow user data to be inserted or appended as shown in Figure 6. In such a case, the invention can create an additional file, and associate it with each file created by the application program. Figure 7 illustrates this association.

The ORIGINAL FILE is that generated by an application program, such as a word-processing program which generated the file "When buying coconuts . . ." as discussed in Figure 2. The invention generates an AUXILIARY FILE, which contains the information of blocks 1 - 6 of Figure 3.

The AUXILIARY FILE of Figure 7 can be created in numerous different ways, and it can conform to numerous different formats. Figure 7 shows a format similar to that of the ORIGINAL FILE, in that both contain SYSTEM HEADERs and APPLICATION PROGRAM HEADERs. Other formats are possible.

For tracking purposes, the invention can give the AUXILIARY FILE an identical name as the ORIGINAL FILE, but with a slight modification. For example, if the ORIGINAL FILE is named "COCONUT.DOC", the invention can name the AUXILIARY HEADER as "COCONUT.AUX".

The invention is expected to find significant use within a system described in the Related Applications. This system is available from AT&T Global Information Solutions Company, Dayton, Ohio, under the name "Continuous Learning System," or CLS. Briefly described, CLS stores resources, such as computer files (which include data files, video files, audio files, and so on) at multiple different computer sites, which can be located throughout the world.

A user, who can be located at any of the sites, or linked to CLS via a communication channel, such as a cellular modem, can obtain a catalogue of all files contained in the system, to which the user is allowed access. The user can retrieve any file desired.

When a user seeks to retrieve the ORIGINAL FILE shown in Figure 7, the invention also fetches the AUXILIARY FILE, reads the necessary information, and proceeds accordingly.

That is, when the ORIGINAL FILE is transmitted or copied, the invention always copies the AUXILIARY FILE, which contains the six blocks of data. Restated, the invention assures that, no matter where the ORIGINAL FILE is transported, the AUXILIARY FILE is always in association with it.

For example, if a user, using CLS, retrieves the ORIGINAL FILE and stores it on a diskette, the invention, without the user's knowledge, also retrieves the AUXILIARY FILE, and stores it on the same diskette. The invention may make the AUXILIARY FILE invisible to the user, so that the user does not see the AUXILIARY FILE when the user reads a directory of the diskette, but the information of the six blocks of Figure 3 are available for the systems of Figure 4.

Recapitulation

Therefore, irrespective of the particular approach taken, such as that shown in Figure 3, 5, 6, or 7, the information of blocks 1-6 of Figure 3 is associated with the ORIGINAL FILE generated by the generating program. The association may be accomplished by writing the data of the six blocks into a separate header of the file, writing the data into an existing header of the file, by creating an AUXILIARY FILE, or by other means.

The six-block information and the ORIGINAL FILE are stored in a repository. If a user seeks to retrieve the ORIGINAL FILE, the invention retrieves it, and also retrieves the six-block information. The six-block information is maintained in association with the ORIGINAL FILE, after retrieval. (That is, within the repository, the ORIGINAL FILE and the AUXILIARY FILE are maintained. Also, at the user's location, the ORIGINAL FILE and the AUXILIARY FILE are now maintained.)

One scenario wherein the invention is to be used is

the following: Assume that a document exists in Italy, written in Italian, and generated by word processing program ABC. The invention creates the six blocks of information shown in Figure 3, and associates them with the file.

If a user in the United States wishes to obtain the file, CLS will retrieve it. CLS maintains a profile for the user, which contains information as to the user's preferred language, preferred word-processing program, and other information.

The invention examines the six headers of the file, and learns that (a) the document is written in Italian, and (b) ABC word processor was used to generate it. (The other four headers will be ignored at present.) The invention examines the profile, and learns that the user prefers English, and prefers word processor XYZ.

The invention then calls appropriate translation programs, which translate the document from Italian into English, and from ABC format into XYZ format. The invention generates a new file, based on the selected file. The new file is ready for loading into the user's XYZ word processing program.

The invention undertakes similar activities with respect to the other four blocks.

Definitional Matters

1. When a file is created by an application program, such as a word-processing program, it contains information which is used by three, or more, agents:

a) the system header, which is used by the operating system (first agent);
b) the application program's header, which is used by the application program (second agent); and
c) the user data, which is used, or created, by the user (third agent).

One or more of these agents are involved in normal activity involving the file. Normal activity includes

a) creating the file (all agents are involved);
b) normal file handling, such as copying, printing, displaying (first agent, or both first and second agents involved); and
c) modifying the file (all agents involved).

Since at least one agent is always involved in normal file activity, these agents can be called "essential agents."

In contrast, the agent which creates, or uses, the six blocks of data shown in Figure 3 is a non-essential agent. The file can be handled in its normal manner, (e. g., created or modified) without the involvement of this agent.

Therefore, one definition of "essential agent" is an agent which is required in the normal file-handling activities of a file, or which is an application program which can load the file.

2. The six blocks actually contain signals which are used by the systems of Figure 4 in calling other programs which transform other data within the file.

Numerous substitutions and modifications can be undertaken without departing from the true spirit and scope of the invention. What is desired to be secured by Letters Patent is the invention as defined in the following claims.

Claims

1. Means for generating a computer file, characterized in that said file comprises a data field and a header in which is stored identification information other than the information required in the presentation of any data stored in said data field on a computer display screen.

2. Means according to claim 1, characterized in that said identification information includes at least one of the following:

i) natural language of the file;
ii) type of encryption of the file;
iii) type of compression of the file;
iv) identity of the program which generated the file;
v) identity of the operating system upon which the generating program ran; and
vi) communication protocol information.

3. A method of processing a computer file, comprising the following steps:

a) maintaining, in association with the file, information indicative of two or more of the following:

i) natural language of the file;
ii) type of encryption of the file;
iii) type of compression of the file;
iv) identity of the program which generated the file;
v) identity of the operating system upon which the generating program ran; and
vi) communication protocol required by the file;

b) reading said information and calling respective programs which process the file in accordance with the information.

FIG. 1

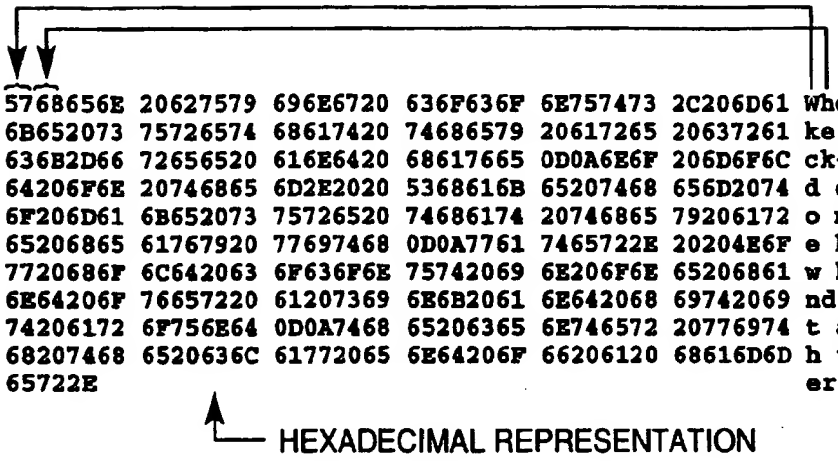

5768656E 20627579 696E6720 636F636F 6E757473 2C206D61 When buying coconuts, ma
6B652073 75726574 68617420 74686579 20617265 20637261 ke surethat they are cra
636B2D66 72656520 616E6420 68617665 0D0A6E6F 206D6F6C ck-free and have no mol
64206F6E 20746865 6D2E2020 5368616B 65207468 656D2074 d on them. Shake them t
6F206D61 6B652073 75726520 74686174 20746865 79206172 o make sure that they ar
65206865 61767920 77697468 0D0A7761 7465722E 20204E6F e heavy with water. No
7720686F 6C642063 6F636F6E 75742069 6E206F6E 65206861 w hold coconut in one ha
6E64206F 76657220 61207369 6E6B2061 6E642068 69742069 nd over a sink and hit i
74206172 6F756E64 0D0A7468 65206365 6E746572 20776974 t around the center wit
68207468 6520636C 61772065 6E64206F 66206120 68616D6D h the claw end of a hamm
65722E er.
HEXADECIMAL REPRESENTATION

FIG. 2

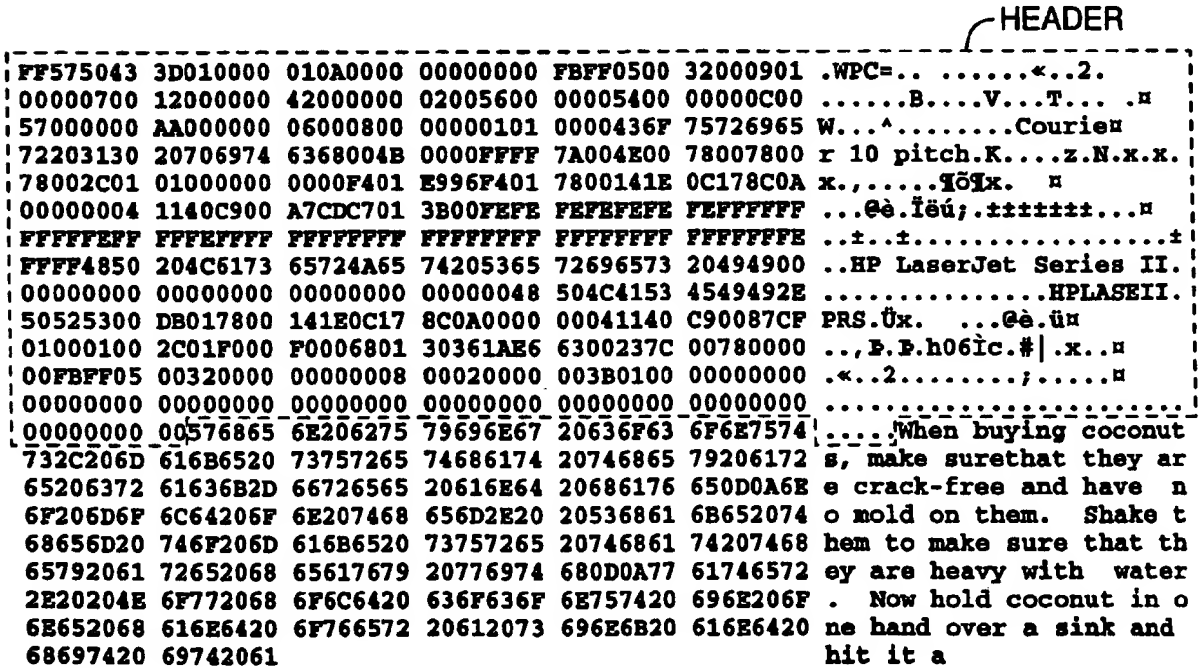
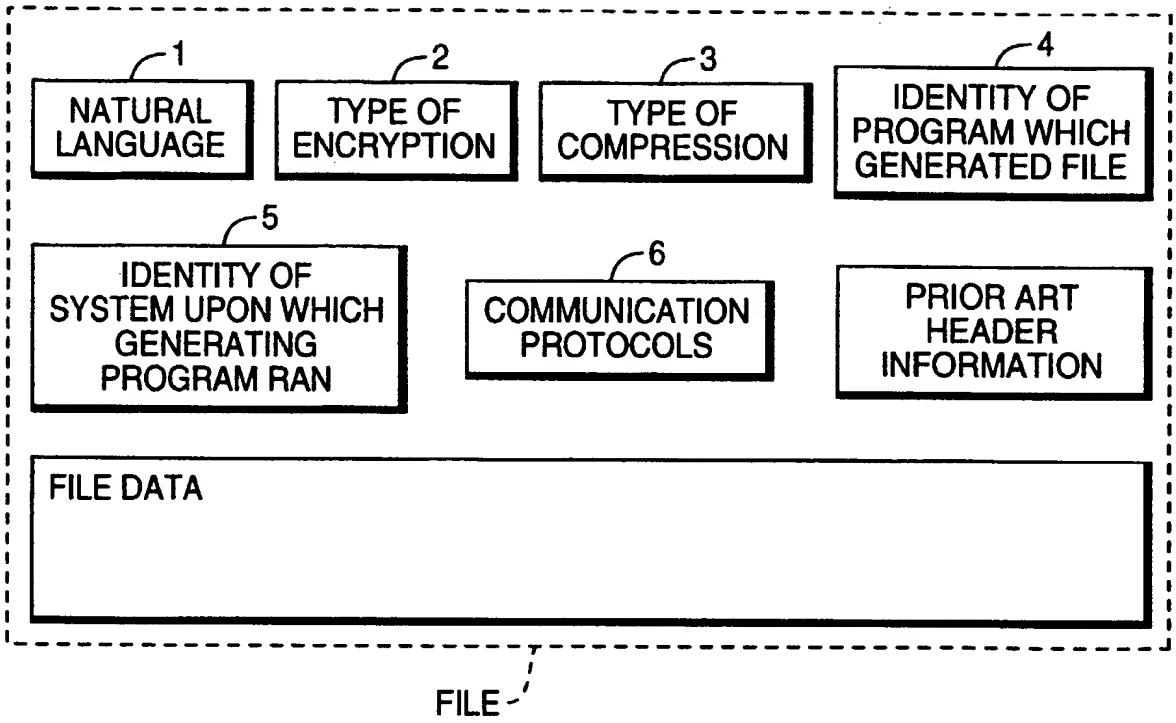

HEADER
FF575043 3D010000 010A0000 00000000 FBFF0500 32000901 .WPC=.. ......«..2.
00000700 12000000 42000000 02005600 00005400 00000C00 ......B....V...T... .¤
57000000 AA000000 06000800 00000101 0000436F 75726965 W...ª........Courie¤
72203130 20706974 6368004B 0000FFFF 7A004E00 78007800 r 10 pitch.K....z.N.x.x.
78002C01 01000000 0000F401 E996F401 7800141E 0C178C0A x.,.....¶õ¶x. ¤
00000004 1140C900 A7CDC701 3B00FEFE FEFEFEFE FEFFFFFF ...@è.Ïëú;.±±±±±±...¤
FFFFFEFF FFFEFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE ..±..±.................±
FFFF4850 204C6173 65724A65 74205365 72696573 20494900 ..HP LaserJet Series II.
00000000 00000000 00000000 00000048 504C4153 4549492E ....HPLASEII.
50525300 DB017800 141E0C17 8C0A0000 00041140 C90087CF PRS.Ûx. ...@è.ü¤
01000100 2C01F000 F0006801 30361AE6 6300237C 00780000 ..,Þ.Þ.h06Ìc.#|.x..¤
00FBFF05 00320000 00000008 00020000 003B0100 00000000 .«..2........;.....¤
00000000 00000000 00000000 00000000 00000000 00000000 ....
00000000 00576865 6E206275 79696E67 20636F63 6F6E7574 ....;When buying coconut
732C206D 616B6520 73757265 74686174 20746865 79206172 s, make surethat they ar
65206372 61636B2D 66726565 20616E64 20686176 650D0A6E e crack-free and have n
6F206D6F 6C64206F 6E207468 656D2E20 20536861 6B652074 o mold on them. Shake t
68656D20 746F206D 616B6520 73757265 20746861 74207468 hem to make sure that th
65792061 72652068 65617679 20776974 680D0A77 61746572 ey are heavy with water
2E20204E 6F772068 6F6C6420 636F636F 6E757420 696E206F . Now hold coconut in o
6E652068 616E6420 6F766572 20612073 696E6B20 616E6420 ne hand over a sink and
68697420 69742061 hit it a

FIG. 3



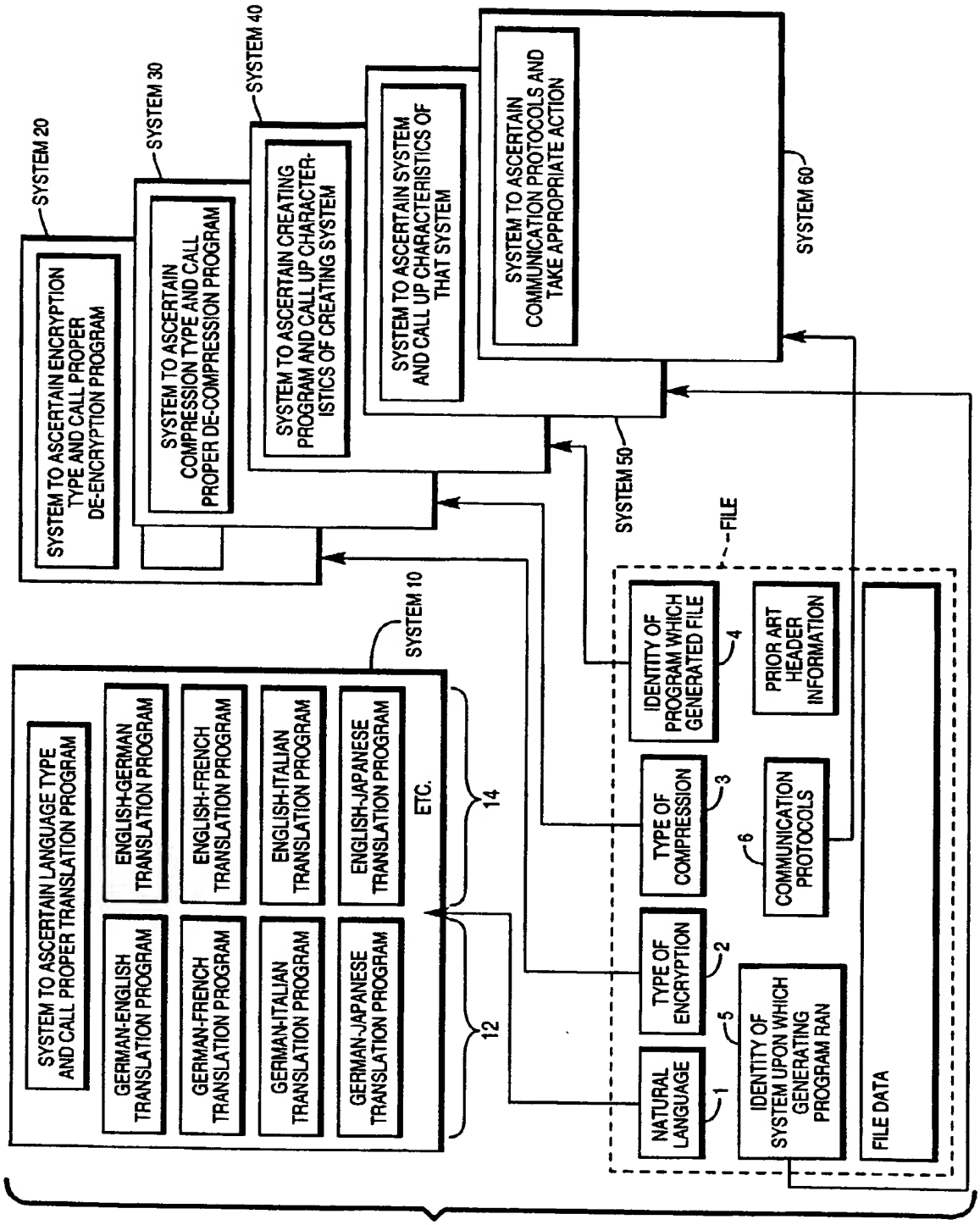

SYSTEM 20
SYSTEM TO ASCERTAIN ENCRYPTION TYPE AND CALL PROPER DE-ENCRYPTION PROGRAM
SYSTEM 30
SYSTEM TO ASCERTAIN COMPRESSION TYPE AND CALL PROPER DE-COMPRESSION PROGRAM
SYSTEM 40
SYSTEM TO ASCERTAIN CREATING PROGRAM AND CALL UP CHARACTER-ISTICS OF CREATING SYSTEM
SYSTEM TO ASCERTAIN SYSTEM AND CALL UP CHARACTERISTICS OF THAT SYSTEM
SYSTEM TO ASCERTAIN COMMUNICATION PROTOCOLS AND TAKE APPROPRIATE ACTION
SYSTEM 60
SYSTEM 50
FILE
SYSTEM 10
SYSTEM TO ASCERTAIN LANGUAGE TYPE AND CALL PROPER TRANSLATION PROGRAM
GERMAN-ENGLISH TRANSLATION PROGRAM
ENGLISH-GERMAN TRANSLATION PROGRAM
GERMAN-FRENCH TRANSLATION PROGRAM
ENGLISH-FRENCH TRANSLATION PROGRAM
GERMAN-ITALIAN TRANSLATION PROGRAM
ENGLISH-ITALIAN TRANSLATION PROGRAM
GERMAN-JAPANESE TRANSLATION PROGRAM
ENGLISH-JAPANESE TRANSLATION PROGRAM
ETC.
12
14
NATURAL LANGUAGE
1
TYPE OF ENCRYPTION
2
TYPE OF COMPRESSION
3
IDENTITY OF PROGRAM WHICH GENERATED FILE
4
IDENTITY OF SYSTEM UPON WHICH GENERATING PROGRAM RAN
5
COMMUNICATION PROTOCOLS
6
PRIOR ART HEADER INFORMATION
FILE DATA

FIG. 4

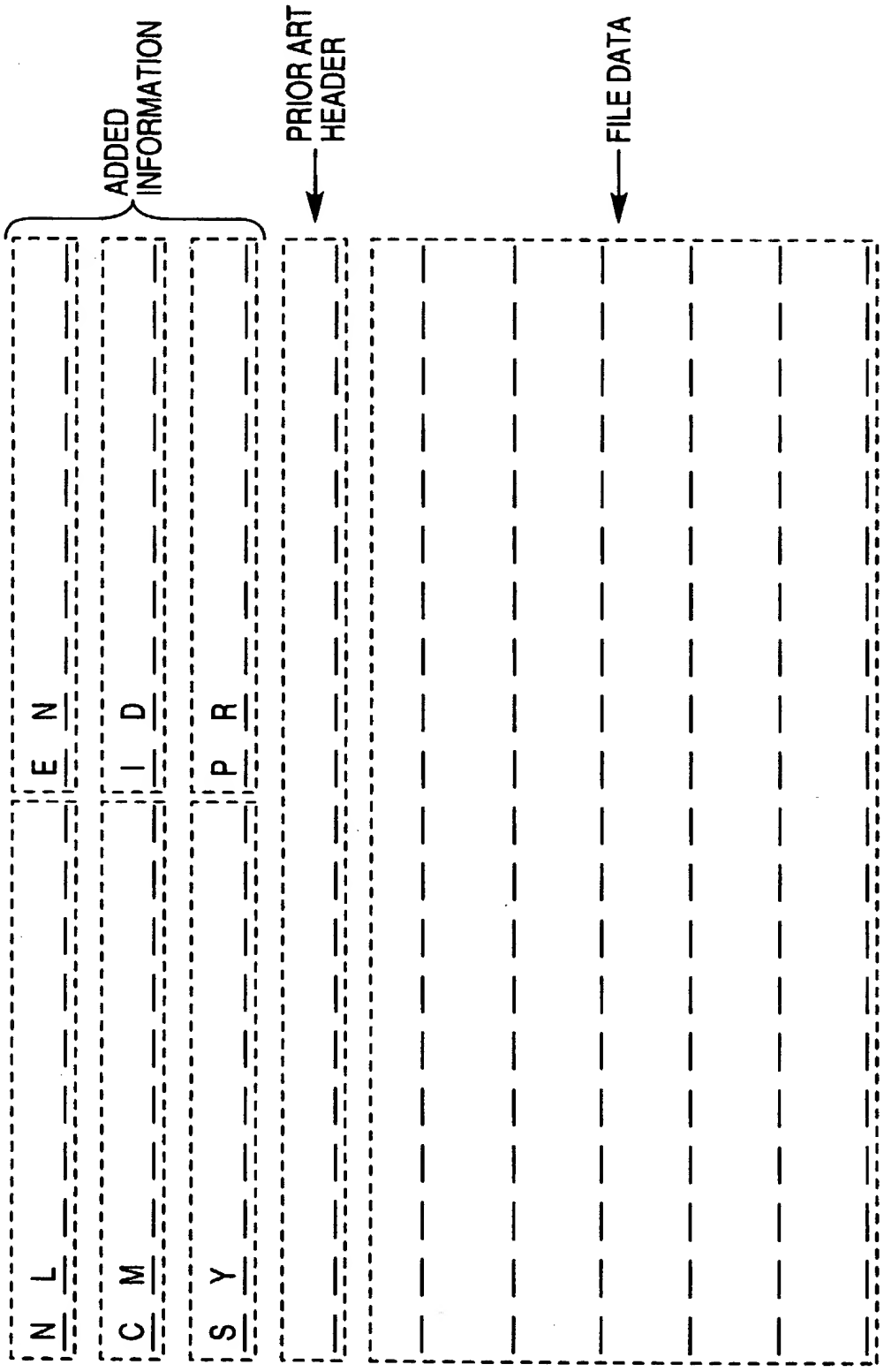
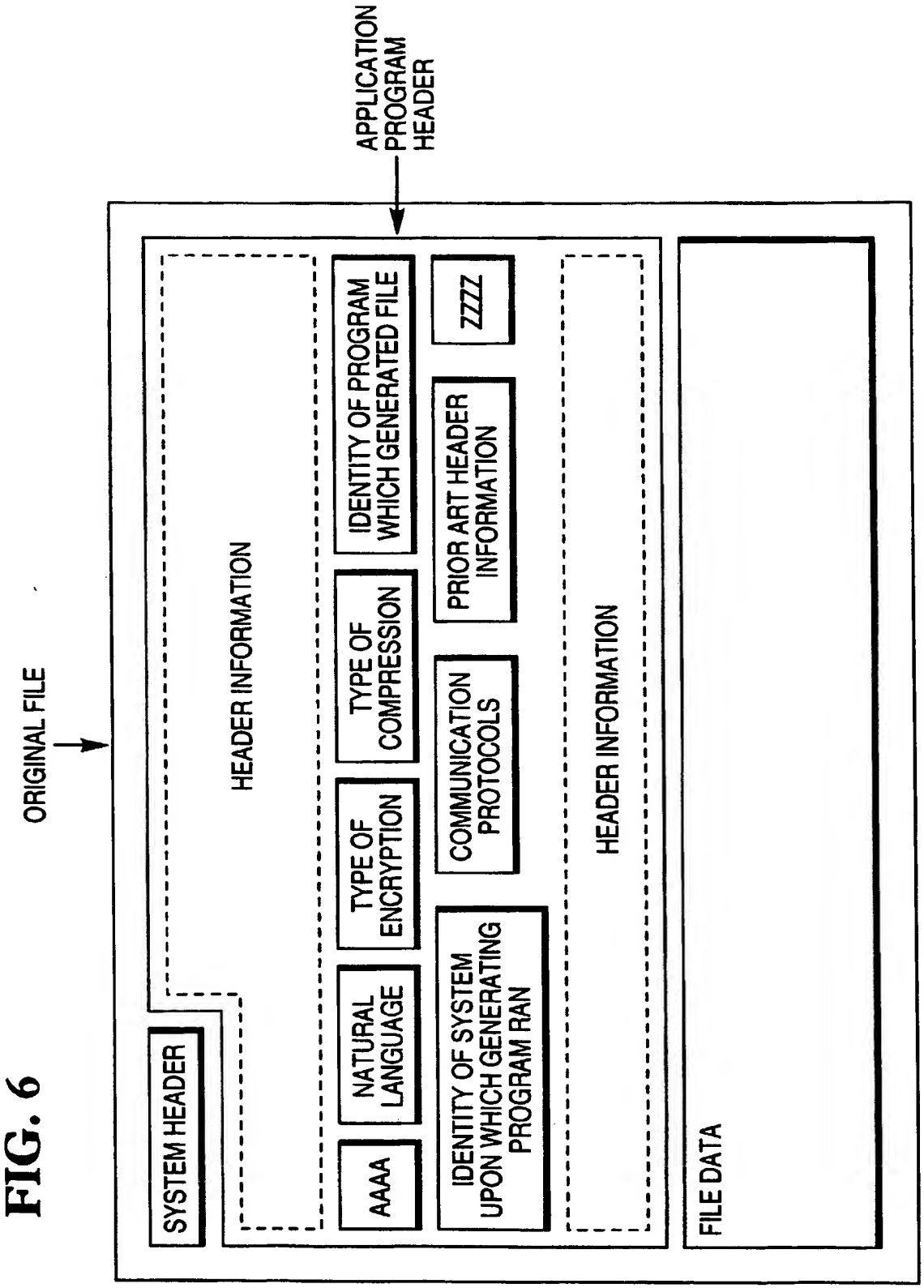


FIG.5

FIG. 6


ORIGINAL FILE
APPLICATION PROGRAM HEADER
SYSTEM HEADER
HEADER INFORMATION
AAAA
NATURAL LANGUAGE
TYPE OF ENCRYPTION
TYPE OF COMPRESSION
IDENTITY OF PROGRAM WHICH GENERATED FILE
IDENTITY OF SYSTEM UPON WHICH GENERATING PROGRAM RAN
COMMUNICATION PROTOCOLS
PRIOR ART HEADER INFORMATION
ZZZZ
HEADER INFORMATION
FILE DATA

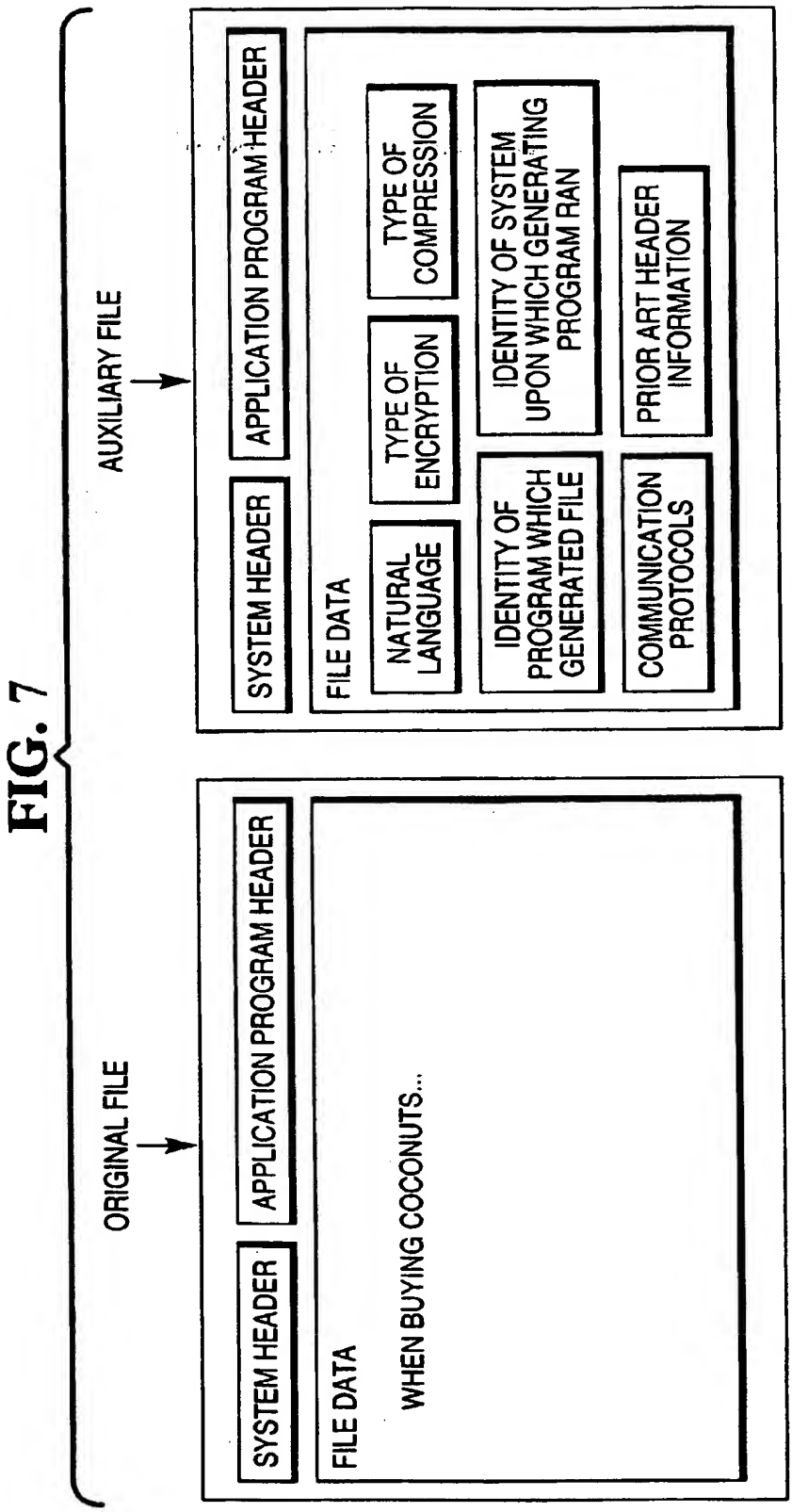


FIG. 7